件(包括头文件)。因此全局变量名称不能与头文件中的变量名相同。

延伸阅读

BJARNE S. C++程序设计语言(第1~3部分)[M]. 王刚，杨巨峰，译. 4版. 北京：机械工业出版社，2016：136-138.

(金靖)

1.2.9.4 递归函数

一个自定义函数既可以被main函数或者其他函数调用，也可以被自身调用。自定义函数在函数体内调用它自身的行为称为递归调用，这种函数称为递归函数。

递归函数首先要确定递归的退出条件，称为递归边界。执行递归函数将反复调用其自身，每调用一次就进入新的一层，传入的参数和函数内的变量不影响上一层的函数。事实上，每一次调用函数，系统给它们分配的内存空间都是独立的。以下是阶乘的递归实现。

```
int factorial(int n) {
    int ans;
    if (n == 0 || n == 1) { //递归边界
        ans = 1;
    } else {
        ans = factorial(n - 1) * n; // 递归调用
    }
    return ans;
}
```

在求阶乘函数中，当$n==0$或$n==1$时，函数才会执行结束，否则就一直调用函数自身。

根据阶乘的定义，$f(n)=f(n-1)\times n$。例如，求5!(5的阶乘)，递归调用过程如下所示：

① 调用factorial(5)，ans=factorial(5-1)×5；

② 调用factorial(4)，ans=factorial(4-1)×4；

③ 调用factorial(3)，ans=factorial(3-1)×3；

④ 调用factorial(2)，ans=factorial(2-1)×2；

⑤ 调用factorial(1)，ans=1。

至此递归已经达到边界，开始自底向上返回调用结果，如下所示：

① ans=1，factorial(2-1)×2=2；

② ans=2，factorial(3-1)×3=6；

③ ans=6，factorial(4-1)×4=24；

④ ans=24，factorial(5-1)×5=120。

因其使用内存空间来实现每一层函数的状态保存，比较消耗内存。在实际使用时要注意递归的层数，否则会导致内存溢出的错误。在大部分 Windows 和 Linux 系统中默认给程序分配的栈大小只有 1MB~8MB 不等，所以需要使用较多栈空间时可以通过编译指令增加操作系统分配的栈空间，以下以 Windows 系统为例。

```
-Wl,--stack=SIZE
```

其中 SIZE 是指定的栈大小，单位是字节。以下编译指令的作用是编译 a.cpp 并为其指定分配 512MB 的栈空间。

```
g++ a.cpp -Wl,--stack=536870912
```

在 Linux 系统中也可以通过 ulimit-s SIZE 命令调大栈空间，其中 SIZE 为一个数字，表示程序可使用的栈空间大小，单位为 kb(或为 unlimited，表示程序可使用的栈空间无限制)。

参考词条

递归法

延伸阅读

KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：73-75.

(金靖)

1.2.10 结构体与联合体

1.2.10.1 结构体

C++可以在已有基本数据类型的基础上自定义新的复合类型，用来存放一组不同类型的数据，称为结构体。结构体在数据结构和算法中应用广泛，常见于图和树的存储等需要自定义复合数据类型的场景。结构体中存放的不同数据，称为结构体的成员变量。如下定义结构体：

```
struct 结构体名 {
    成员类型名 成员变量名;
    ...
};
```

结构体变量可以在定义时使用“{}”进行初始化，再依次对结构体变量的各个成员变量赋值，如下：

```
结构体类型 结构体变量={成员变量1初始值,成员变量2初始值,…};
```

两个同类型的结构体变量，可以直接使用“=”运算符赋值。用“结构体变量名.成员变量名”的方式，访问结构体变量的成员变量。

一个结构体变量占用的空间是各成员变量占用的空间之和。

参考词条

类的概念及简单应用

延伸阅读

BJARNE S. C++程序设计语言(第1~3部分)[M]. 王刚，杨巨峰，译. 4版. 北京：机械工业出版社，2016：173-186.

（金靖）

1.2.10.2 联合体

联合体“union”是一种构造类型的数据结构，在一个联合体中可以定义多种不同的数据类型，这点与结构体相似。如下定义联合体：

```
union 联合体名 {
    成员类型名 成员变量名;
    …
};
```

联合体变量可以存入其定义的任何一种数据类型，各成员变量共享一段内存空间，即每次只能赋一种值，赋入新值则覆盖旧值。一个联合体变量占用的空间等于各成员中最大的变量所占用的空间。

（金靖）

1.2.11 指针类型

1.2.11.1 指针

指针是一种特殊的变量，存放的是所指向对象在内存中的起始地址。在C++中，可以用指针变量来存放内存地址。

对于类型 T，用 $T*$ 表示“指向 T 的指针”的类型，即 $T*$ 类型的变量存放的是 T 类型的变量的地址。

为了赋值一个指针，就需要取出一个变量的地址，& 为取地址运算符，$\&a$ 表示变量 a 的地址。

为了调用一个指针，需要取出它指向的变量的值，$*$ 为间接取值运算符，$*p$ 表示

指针 p 所指向的变量的值。

```
char c = 'a';
char *p = &c; //p 存放 c 的地址
char c2 = *p; //c2='a'
```

对于指向数组的指针，如果数组类型为 $T[s_1][s_2]\cdots[s_n]$，则指向这种数组的指针类型为 $T(*)[s_1][s_2]\cdots[s_n]$。注意区分 `int *a[10]`和 `int (*a)[10]`。前者表示定义一个长度为 10 的数组，每个元素类型为 int*，而后者表示定义一个指向一个长度为 10 的整型数组的指针。具体代码如下。

```
int a[10] = {1,2,3};
int (*p)[10] = &a; //让 p 指向 a
cout << (*p)[0] << " " << (*p)[1] << endl; //运行结果:1 2
```

对于指向函数的指针，如果函数返回值类型为 T，参数类型为 $T_1,T_2,\cdots,T_n$，那么用 $T(*)(T_1,T_2,\cdots,T_n)$表示指向这种函数的指针的类型。具体代码如下。

```
int calc(int a,int b) {
    return a + b;
}
int main() {
    int (*p)(int,int); //定义一个类型为 int(*)(int,int)的指针 p,指向类型为 int
                       (int,int) 的函数
    p = calc;
    cout << p(1,2) << endl; //调用 p 指向的函数
    return 0;
}
```

声明指针时应将其赋值，否则对未赋值的指针取值会访问不确定的内存空间，从而引起未知错误。

延伸阅读

[1] KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：79.

[2] BJARNE S. C++程序设计语言(第 1~3 部分)[M]. 王刚，杨巨峰，译. 4 版. 北京：机械工业出版社，2016：148-150.

（金靖）

1.2.11.2 基于指针的数组访问

C++中的一维数组名可以看作指向首个元素的指针，即 `a=&a[0]`。

若指针 p 指向数组 a 中某个元素，则 $p+1$ 指向它的下一个元素，$p+i$ 就指向它后面

第 i 个元素，$p-i$ 指向它前面第 i 个元素。因此，如果 p 指向 $a[0]$，那么 $p+i$ 就指向 $a[i]$，有 p+i=&a[i]，*(p+i)=a[i]=*(a+i)。赋值 p=&a[0]可以简写成 p=a。

数组和指针也有一些差异。对于类型为 T 的数组 a，sizeof(a)返回的是数组长度 * sizeof(T)，而 sizeof(&a[0])返回的是一个指针所占用的空间。而且若 p 为指针，则 p=a 和 p++是合法的，但是 a=p 和 a++是不合法的。

当数组作为函数的参数时，实际传输的是数组首个元素的地址，将数组以指针形式传输过去。

延伸阅读

KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：83-86.

（金靖）

1.2.11.3 字符指针

字符指针是指向字符类型的指针，与普通指针的使用方法基本一致。

可以将字符串的首地址赋给字符指针，使其指向字符串。若 $s1$ 为 const char* 型指针，可以直接用字符串字面值常量赋值，s1=“Hello”；合法。但若 $s2$ 为字符数组，则语句 s2=“Hello”；不合法。

对于 string 类的 str 变量，可以用 str. c_str()将其转化为 const char* 的类型，也可以直接用 char* 和 const char* 的变量给其赋值。

参考词条

字符串的处理

延伸阅读

[1] KERNIHAN B W，RITCHIE D M. C 程序设计语言[M]. 徐宝文，李志，译. 2 版. 北京：机械工业出版社，2004：89-90.

[2] BJARNE S. C++程序设计语言(第 1~3 部分)[M]. 王刚，杨巨峰，译. 4 版. 北京：机械工业出版社，2016：77.

（金靖）

1.2.11.4 指向结构体的指针

指向一个结构体变量的指针，就是该变量所占据的内存段的起始地址。

对于一个结构体指针 p，要调用其成员对象 a，可以写成(*p).a，或者更简单地写成 p->a，注意不能写成*p. a。

当函数变量参数要传递结构体时，如果按值传递，就要复制整个结构体，效率不高，一般可以使用结构体指针。